

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 05-197547

(43)Date of publication of application : 06.08.1993

(51)Int.Cl.

G06F 9/38

G06F 9/38

(21)Application number : 04-155190

(71)Applicant : TOSHIBA CORP

(22)Date of filing : 15.06.1992

(72)Inventor : ISHIKAWA TEI

(30)Priority

Priority number : 03286100

Priority date : 31.10.1991

Priority country : JP

03286224

31.10.1991

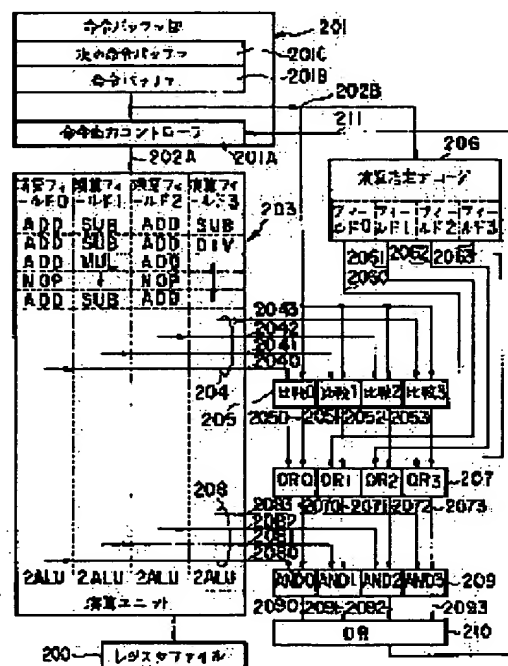
JP

(54) VLW TYPE ARITHMETIC PROCESSOR

(57)Abstract:

PURPOSE: To speed up arithmetic processing by executing following instruction words in parallel without waiting the completion of the arithmetic of precedent instruction words unless a register number is referred to for the arithmetic of the following instruction words.

CONSTITUTION: A comparator 205 compares the number of a register expected to store the execution result of an instruction, which begins to be executed earlier than an instruction word to currently be executed, with the number of a register to be read out by the instruction to currently be executed. When there is a matching register number, the instruction to currently be executed is held in a waiting state by an instruction controller 201A once it is confirmed that the contents of the register having the register number can not be used yet. In other cases, the following instructions which do not refer to the execution result of the instruction being executed are executed in parallel even while the instruction which requires plural cycles for its processing is being executed, thus speed up the processing of plural instruction words.



LEGAL STATUS

[Date of request for examination]

09.03.1998

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	2928684
[Date of registration]	14.05.1999
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

Copyright (C); 1998,2003 Japan Patent Office

(19)日本国特許庁(JP)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平5-197547

(43)公開日 平成5年(1993)8月6日

(51)Int.Cl.⁵

G 0 6 F 9/38

識別記号

3 1 0 J

庁内整理番号

9290-5B

3 7 0 X 9290-5B

F I

技術表示箇所

審査請求 未請求 請求項の数1(全 15 頁)

(21)出願番号 特願平4-155190

(22)出願日 平成4年(1992)6月15日

(31)優先権主張番号 特願平3-286100

(32)優先日 平3(1991)10月31日

(33)優先権主張国 日本(JP)

(31)優先権主張番号 特願平3-286224

(32)優先日 平3(1991)10月31日

(33)優先権主張国 日本(JP)

(71)出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72)発明者 石川 禎

東京都府中市東芝町1番地 株式会社東芝

府中工場内

(74)代理人 弁理士 鈴江 武彦

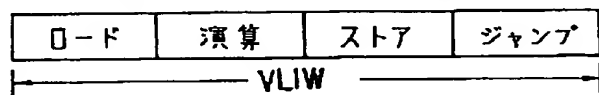
(54)【発明の名称】 VLIW型演算処理装置

(57)【要約】

【目的】VLIW型演算処理装置における複数命令並列処理の速度改善。

【構成】先行する演算が実行中のため内容が確定していないレジスタ(オペランド)を後続の命令語が参照しようとする場合、あるいは後続命令語がまだ演算実行中の演算器を使用しようとする場合には、演算が完了して演算結果が確定するまで後続命令語の実行を待たせ、これらの場合以外では先行する命令語の演算が実行中であっても後続命令語を実行する制御を導入して、並列処理の効率を改善する。

フィールド0 フィールド1 フィールド2 フィールド3



【特許請求の範囲】

【請求項1】 複数の命令語を含むVLIWを格納する格納手段と；前記VLIWに含まれる命令語数に対応した数の複数フィールドからなるパイプラインを有し、これらのパイプラインで前記VLIWに含まれる命令を独立に実行する実行手段と；前記実行手段により実行される各命令語のオペランドにより参照されるハードウェアリソースと；前記実行手段のあるフィールドで現在実行中の命令の実行結果を格納するための第1オペランドと、前記実行手段の各フィールドで現在実行しようとする命令により参照される第2オペランドとを比較し、前記第2オペランドが前記第1オペランドと一致するものを含む場合に、一致信号を提供する提供手段と；前記実行手段の各フィールドで命令実行が終了していない場合に、フィールド毎に未了信号を発生する発生手段と；前記提供手段からの前記一致信号と前記発生手段からの前記未了信号との論理積を前記実行手段の各フィールド毎にとり、何れかのフィールドが前記論理積に対して真をもたらず場合に命令取り込み禁止信号を出力し、全てのフィールドが前記論理積に対して疑をもたらず場合に命令取り込み信号を出力する出力手段と；前記出力手段が前記命令取り込み信号を出力するときは前記格納手段から前記実行手段の前記パイプラインへ前記VLIWに含まれる1以上の命令語を同時に転送し、前記出力手段が前記命令取り込み禁止信号を出力するときは前記格納手段から前記実行手段への前記VLIWの命令語転送を停止する手段とを具備したことを特徴とするVLIW型演算処理装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 この発明は、ストアードプログラム型デジタル計算機におけるVLIW (Very Long Instruction Word) 型演算処理装置に関する。

【0002】

【従来の技術】 マイクロプロセサの処理性能は、その動作周波数、1命令当たりの所要サイクル数CPI (Cycles per Instruction)、およびプログラム実行に必要な命令数の3つの要因で決まる。すなわち、処理性能を上げたければ、動作周波数を上げ、CPIを下げ、プログラム実行命令数を減らすことが必要となる。

【0003】 上記3要素のうち、半導体製造技術の進歩によりマイクロプロセサチップの動作周波数は年々上がっているが、その時々で入手可能なチップの動作周波数には上限がある。

【0004】 また、マイクロプロセサにはCISC (Complexed Instruction Set Computer) 型とRISC (Reduced Instruction Set Computer) 型

(およびこれらの折衷型) があり、RISCチップならCISCチップよりもCPIを小さくできるが、それでもCPIは1までにしか下らない。プログラム実行命令数はプログラマの能力（あるいはコンパイラの性能）に依存するが、これを減らすにも限界がある。

【0005】 このような状況を背景として、VLIWアーキテクチャとスーパースカラアーキテクチャが生まれてきた。これらのアーキテクチャを採用したプロセサでは、プログラム中から並列に実行できる命令を見つけて、複数の処理を並列に実行する。すなわち、複数の命令を同時にデコードしてそれらを実行ユニットに送り、同時に処理を行なう。このような並列処理により、CPIを1以下に下げることが可能となる。

【0006】 VLIW型プロセサおよびスーパースカラ型プロセサは複数のパイプラインを内臓しており、このパイプラインにより複数の処理を同時に実行できる点では同じであるが、両者は以下の点で相違する。

(1) 命令ストリームのスケジューリング方法の相違

【0007】 スーパースカラでは、プロセサの命令デコーダが並列に実行できる命令を見つけ、命令実行段階で動的にスケジューリングを行なう。このため、並列性の検出はフェッチした命令からしか行なえない。

【0008】 これに対し、VLIWでは、コンパイル時（あるいはアセンブラコーディング時）にプログラムをスケジューリングする。そのため、スーパースカラに比べコンパイラ（あるいはプログラマ）の負担は大きくなるが、VLIWではハードウェア（命令デコーダ）が簡単になる。

(2) 命令フォーマットの相違

【0009】 一般に、スーパースカラは従来のRISCプロセサと同じ32ビット長の命令フォーマットを持っている。そのデコーダは、フェッチした命令から並列に実行できる命令を選び、パイプラインに投入する。32ビット長の命令フォーマットをサポートするスーパースカラプロセサでは従来のRISCプロセサの命令フォーマットをそのまま利用することができ、オブジェクトコードレベルで、既存のRISCとソフトウェアの互換性を維持できる。

【0010】 これに対し、VLIWは、一般に32ビットよりも長い命令フォーマットを持っており、1命令で複数の処理を指定することができる。そのため、既存のRISCとソフトウェア互換性を維持することはできないが、スーパースカラよりも高い性能を出すことができる。

【0011】 ところで、パイプライン構造を持つプロセサの場合、1つの命令は複数のステージに分けられて実行される。1つのステージは原則として1クロックで実行されるため、パイプラインが順調に動作すれば全ての命令は見かけ上1クロックで実行されることになる。しかし、実際には様々な原因でパイプラインは乱れプロセ

サの実質的な処理速度は低下する。これらの原因はハザードと呼ばれる。このハザードをハードウェアで検出して自動的にパイプラインを遅らせる機構をインターロックという。

【0012】この発明は、スーパースカラではなくVLIWアーキテクチャに基づく並列処理システムに関するものである。とくに、上記インターロックの機能を改良し、パイプラインの遅れを最小限に押さえ処理の高速化を図ったVLIW型演算処理装置に関する。

【0013】一般に、VLIW型演算処理装置では複数の演算を指定できる固定長の命令語を実行して行くが、1つの命令語中に指定された複数の演算は、少なくともソフトウェア（プログラム）からは同時に実行されているように見えることが必要である。

【0014】VLIW型演算処理装置では、その処理能力の向上のために浮動小数点演算などもサポートされている。この浮動小数点演算は、種類によっては（例えば除算では）その処理に数サイクルを要する。

【0015】ところが、従来のVLIW型演算処理装置では、同時に実行されるべく指定された1命令中の複数演算は、ソフトウェアに対しては、あくまで同時に実行されるように見せなければならない。そのために、同時に指定された複数演算中に処理時間が長くなるものがあった場合、それ以外の演算が先に終了していたとしても、この長時間演算処理が完了するまでは、演算が終了した演算器に次の演算処理をさせず、遊ばせていた。

【0016】

【発明が解決しようとする課題】上述のように、同時に実行されるべく指定された1命令語中の複数演算はソフトウェア上はあくまで同時に実行されていることになっていなければならないために、従来のVLIW型演算処理装置では、先行する命令語に含まれる演算処理が全て終了するまでは次の命令語を実行しないようにしている。このため、同時に指定された複数演算中に処理時間の長くなるものがあった場合、それが完了するまでは、他の演算が既に終了していたとしても、演算の終了している演算器に対して次の命令を実行させずに待たせたまま遊ばせることになる。すると並列演算時間が長くなり、VLIWアーキテクチャの並列演算処理能力を十分に活用できない。

【0017】この発明は、上記事情に鑑みなされたもので、ソフトウェアに対しては演算の実行順序を保証しつつ、先行して起動された時間のかかる演算が実行中であっても、できるかぎり後続の命令語を並列に実行できるようにし、もって並列処理用演算器の稼働率を向上させ、演算処理装置の全体的な処理能力を改善したVLIW型演算処理装置を提供することを目的とする。

【0018】

【課題を解決するための手段】この発明のVLIW型演算処理装置は、複数の命令語を含むVLIWを格納する

命令バッファ（201）と；

【0019】前記VLIWに含まれる命令語数に対応した数の複数フィールドからなるパイプラインを有し、これらのパイプラインで前記VLIWに含まれる命令を独立に実行する演算ユニット（203）と；前記演算ユニットにより実行される各命令語のオペランド（R_j）により参照されるハードウェアリソース（200）と；

【0020】前記演算ユニットのあるフィールド（i=1）で現在実行中の命令（MUL）の実行結果を格納するための第1オペランド（R₄）と、前記演算ユニットの各フィールド（i=0~3）で現在実行しようとする命令（Add；i=0, 2）により参照される第2オペランド（R₁~R₄）とを比較し、前記第2オペランド（R₁~R₄）が前記第1オペランド（R₄）と一致するもの（R₄）を含む場合に、一致信号を提供する比較器（205）と；前記演算ユニットの各フィールド（i）で命令実行が終了していない場合に、フィールド毎に未了信号を発生する手段（208）と；

【0021】前記比較器からの前記一致信号と前記発生手段からの前記未了信号との論理積を前記演算ユニットの各フィールド（i）毎にとり、何れかのフィールド（i）が前記論理積に対して真をもたらず場合に命令取り込み禁止信号を出力し、全てのフィールド（i）が前記論理積に対して疑をもたらず場合に命令取り込み信号を出力する論理回路（209, 210）と；

【0022】前記論理回路が前記命令取り込み信号を出力するときは前記命令バッファから前記演算ユニットの前記パイプラインへ前記VLIWに含まれる1以上の命令語を同時に転送し、前記論理回路が前記命令取り込み禁止信号を出力するときは前記命令バッファから前記演算ユニットへの前記VLIWの命令語転送を停止する命令出力コントローラ（201A）とを具備している。

【0023】

【作用】この発明のVLIW型演算処理装置では、現在実行しようとしている命令語より先に実行が開始された命令が実行中であり、その命令の実行結果を格納する予定のレジスタ番号と現在実行しようとしている命令によって読出されるレジスタ番号とを比較する。そして一致するレジスタ番号がある場合に当該レジスタ番号のレジスタの内容がまだ使用不可能であることが認識されると、現在実行しようとしている命令を待たせる。それ以外の場合では、処理に複数サイクルかかる命令が実行中であっても、その命令の実行結果を参照しない後続の命令は並列実行させて、複数の命令語の処理の高速化を図る。

【0024】

【実施例】図1は、フィールド0~3それぞれに4つの32ビット長命令語（ロード、演算、ストア、ジャンプ）を配した128ビットのVLIWの一例を示す。この演算命令には、加算（ADD）、減算（SUB）、乗

算(MUL)、除算(DIV)、無処理(NOP)等が含まれる。

【0025】図2は、この発明が適用されない場合のVLW並列演算処理の一例を示す。この例では、フィールド3で除算($R3/R4$)を4クロックかけて実行中にフィールド0~2の実行ユニットが3クロック分遊んでしまっており、フィールド1で乗算($R1 * R2$)を2クロックかけて実行中にフィールド0, 2, 3の実行ユニットが1クロック分遊んでしまっている。この遊びがあるために、図2で例示する演算処理の完了には、都合8クロックかかっている。

【0026】図3は、この発明が適用された場合のVLW並列演算処理の一例を示す。この例では、フィールド3で4クロック除算($R3/R4$)を実行中に、フィールド1で2クロック乗算($R1 * R2$)と2度の1クロック減算($R1 - R2$)を実行し、フィールド0とフィールド2で2度の1クロック加算($R1 + R2$; $R3 + R4$)とタイミング合わせのための1クロックNOPを1度実行している。図3に例示する演算処理では、除算・乗算の処理中にパイプライン中の実行ユニットが遊ぶ(NOP実行)割合が少ないので、都合5クロックで図2と同等の演算処理を終了している。図6は、図3で例示したような並列演算処理がハードウェア中でどのように進行するかを説明するフローチャートである。

【0027】まず、例えば図3の命令語1がハードウェア(並列演算実行ユニット)にフェッチされたとする(ST100)。このフェッチにより、フィールド0~3の4本のパイプラインには命令語1のADD, SUB, ADD, SUBが一括して(同期して)投入される。

【0028】各フィールド0~3では、投入された命令(ADD, SUB, ADD, SUB)のオペランド(命令語が参照するアドレスまたはデータのこと。ここでは各パイプラインで参照されるレジスタファイルのレジスタ番号 Rj ; j は例えば1~8)が使用可能かどうかチェックされる(ST110~ST113)。今は演算を開始し始めたばかりであり、投入された命令(ADD, SUB, ADD, SUB)が要求する各フィールド0~3のオペランド($R1 \sim R4$)は全て使用可能である(ST114~ST117, イエス)。

【0029】フィールド0~3のオペランドが全て使用可能となると(ST120, イエス)、各フィールド0~3の演算に必要なハードウェアリソース(レジスタファイル中で現在使用可能なレジスタ)があるかどうかチェックされる(ST130~ST133)。今は演算を開始し始めたばかりであり、ハードウェアのリソースは全て使用可能である(ST134~ST137, イエス)。

【0030】フィールド0~3のハードウェアリソースが全て使用可能となれば(ST140, イエス)、各フ

ィールド0~3で、投入された命令(ADD, SUB, ADD, SUB)の処理が同時に開始する(ST150~ST153)。以上の処理により、図3の命令語1に対する処理が1クロックで完了する。

【0031】次に、図3の命令語2がハードウェアにフェッチされる(ST100)。このフェッチにより、フィールド0~3の4本のパイプラインには命令語2のADD, SUB, ADD, DIVが投入される。

【0032】各フィールド0~3では、投入された命令(ADD, SUB, ADD, DIV)のオペランドが使用可能かどうかチェックされる(ST110~ST113)。命令語1は1クロックで同時処理完了しており、かつフィールド3の命令DIVが参照するオペランド(レジスタR8)はまだ未使用なので、投入された命令(ADD, SUB, ADD, DIV)が要求する各フィールド0~3のオペランド($R1 \sim R4$, R8)は全て使用可能である(ST114~ST117, イエス)。

【0033】フィールド0~3のオペランドが全て使用可能となると(ST120, イエス)、各フィールド0~3の演算に必要なハードウェアリソースがあるかどうかチェックされる(ST130~ST133)。投入された命令(ADD, SUB, ADD, DIV)が要求するハードウェアのリソースはいずれも他の命令により使用中でないで、全て使用可能である(ST134~ST137, イエス)。

【0034】フィールド0~3のハードウェアリソースが全て使用可能となれば(ST140, イエス)、各フィールド0~3で、投入された命令(ADD, SUB, ADD, DIV)の処理が同時に開始する(ST150~ST153)。以上のようにして、図3の命令語2が処理される。

【0035】次に、図3の命令語3がハードウェアにフェッチされる(ST100)。このフェッチにより、フィールド0~2の3本のパイプラインには命令語3のADD, MUL, ADDが投入される。(図3の処理プログラムを生成したコンパイラは、命令語2でフィールド3に投入した命令DIVが時間のかかる処理であることを知っているため、この時点ではフィールド3に新たな命令は投入されない。)

【0036】各フィールド0~2では、投入された命令(ADD, MUL, ADD)のオペランドが使用可能かどうかチェックされる(ST110~ST113)。ここで、フィールド3はまだ命令語2のDIVを処理中であり、この命令DIVの処理はフィールド3の2つのALU内で行なわれている。このため、フィールド3のオペランド($R3$, $R4$, $R8$)はまだ塞がっており、使用可能である(ST117, イエス)。また、命令語2のフィールド0~2は1クロックで同時処理完了しているので、投入された命令(ADD, MUL, ADD)が要求する各フィールド0~2のオペランド($R1 \sim R8$)

4)も全て使用可能である(ST114~ST116, イエス)。

【0037】フィールド0~3のオペランドが全て使用可能となると(ST120, イエス)、各フィールド0~3の演算に必要なハードウェアリソースがあるかどうかチェックされる(ST130~ST133)。

【0038】この場合、フィールド1で処理中の命令MULはリソース(レジスタR4)を使っており、このリソース(レジスタR4)をフィールド2の命令ADDが参照しているため、フィールド2の演算に必要なハードウェアリソースはない(ST136, ノー)。そこで、このリソース(レジスタR4)が空くまで(つまりフィールド1の命令MULの実行が終わるまで)、フィールド2は何もしない命令NOPを挟んで待つ(ST132, ST136のループ)。

【0039】このとき、フィールド2の命令ADDはフィールド0の命令ADDが使うリソース(レジスタR3)も参照している。このため、フィールド1の命令MULの実行が終わるまでフィールド2が命令NOP(1クロック)を実行して待っている間、フィールド0も命令NOP(1クロック)を実行して待つ(ST130, ST134のループ)。

【0040】フィールド0、2に命令NOPを挟むことによりフィールド0~3のハードウェアリソースが全て使用可能となり(ST140, イエス)、各フィールド0~3で、投入された命令(ADD, MUL, ADD)および実行中の命令(DIV)の並列処理が、2クロックかけて同時進行する(ST150~ST153)。以上のようにして、図3の命令語3が処理される。

【0041】次に、図3の命令語4がハードウェアにフェッチされる(ST100)。このフェッチにより、フィールド0~2の3本のパイプラインには命令語4のADD, SUB, ADDが投入される。(図3の処理プログラムを生成したコンパイラは、命令語2でフィールド3に投入した命令DIVが時間の係る処理であることを知っているため、ここでもフィールド3に新たな命令は投入されない。)

【0042】各フィールド0~2では、投入された命令(ADD, SUB, ADD)のオペランドが使用可能かどうかチェックされる(ST110~ST113)。命令語3のフィールド0~2は2クロックで同時処理完了しているので、投入された命令(ADD, SUB, ADD)が要求する各フィールド0~2のオペランド(R1~R4)は全て使用可能である(ST114~ST116, イエス)。また、フィールド3はまだ命令語2のDIVをフィールド3のALUで処理中であり、この命令DIVに対するオペランド(R3, R4, R8)は使用可能である(ST117, イエス)。

【0043】フィールド0~3のオペランドが全て使用可能となると(ST120, イエス)、各フィールド0

~3の演算に必要なハードウェアリソースがあるかどうかチェックされる(ST130~ST133)。投入された命令(ADD, SUB, ADD)および実行中の命令(DIV)が要求するハードウェアのリソースは塞がっていないので、全て使用可能である(ST134~ST137, イエス)。

【0044】フィールド0~3のハードウェアリソースが全て使用可能となれば(ST140, イエス)、各フィールド0~3で、投入された命令(ADD, SUB, ADD)および実行中の命令(DIV)の並列処理が同時進行する(ST150~ST153)。以上のようにして、図3の命令語4が処理される。

【0045】図7は、この発明の一実施例に係るVLIW並列演算処理装置の構成を説明するブロック図である。この実施例装置は、レジスタファイル200と、命令バッファ部201と、演算ユニット203と、レジスタ比較回路205と、演算指定デコーダ206と、オア回路207と、アンド回路209と、オア回路210とを備えている。

【0046】命令バッファ部201は、図1に示すような複数演算を包含する命令語群からなるVLIWを格納する。命令バッファ部201は、後述する信号211によって命令バッファ201B内のVLIWを出力する命令出力コントローラ201Aと、命令バッファ201B内の命令の次に実行される命令を格納するバッファ201Cを含んでいる。

【0047】命令バッファ部201に格納されたVLIWは、信号線202Aを介して演算ユニット203に転送される。演算ユニット203は、転送されたVLIWに含まれる複数演算を、レジスタファイル200を用いて、4本のパイプライン(フィールド0~3)により並列処理する。この処理のために、これらのパイプラインはそれぞれ独自の演算器(ALU)を複数台(例えば2台)備えている。

【0048】演算ユニット203で現在実行中の演算の結果が格納されるところのレジスタファイル200のレジスタ番号Rj(jは例えば1~8)は、信号線204i(i=0, 1, 2, 3)を介して、レジスタ比較回路205に与えられる。このレジスタ比較回路205には、信号線202Bを介して、次に実行しようとするVLIWによって参照されるレジスタ番号も与えられる。レジスタ比較回路205は、前者のレジスタ番号と後者のレジスタ番号をと比較し、両者が一致するかどうかを調べる。

【0049】すなわち、レジスタ比較回路205は4つの比較ブロック0~3により構成され、これらのブロック0~3は演算ユニット203のフィールド0~3にそれぞれ対応している。各ブロックi(i=0~3)では、そのフィールドiで現在実行中の演算の結果が格納されるレジスタ番号Rj(j=1~8)と、次の命令語

によって参照される全てのレジスタ番号との一致が調べられる。

【0050】演算指定デコーダ206には、命令バッファ202Bから、信号線202Bを介して、次に実行しようとしている命令語が与えられる。デコーダ206は与えられた命令語から、NOP以外の演算が指定されているフィールドiを検出する。そのようなフィールドiが検出されたなら、デコーダ206はこのフィールドiの演算器が空くまで次の命令語の実行を待たせる信号206iを出力する。

【0051】オア回路207は、レジスタ比較回路205の各ブロックiごとの出力205iと演算指定デコーダ206の各フィールドiごとの出力206iとの論理和を取る。またアンド回路209は、オア回路207の各ORゲートiごとの出力207iと信号208iとの論理和を取る。信号208iは、演算ユニット203の各フィールドiから取り出されるもので、各フィールドiごとに現在実行中の演算がまだ完了していないことを示す。

【0052】アンド回路209の各ANDゲートiごとの出力209iはオア回路210に入力される。オア回路210の出力は、信号線211を介して、命令出力コントローラ201Aに送られる。

【0053】次に、図8および図9を参照して、上記構成のVLIW型演算処理装置の動作を説明する。(この動作はマイクロプログラムではなくハードウェアロジックにより実行される。)

【0054】この実施例のVLIW型演算処理装置は、次のような特徴を持つ。すなわち、原則として複数命令語の並列処理を可とするが、現在実行中の命令語の演算結果が格納されるレジスタファイル200のレジスタ番号(オペランド)と、次に実行しようとしている命令語によって参照されるレジスタ番号(オペランド)とが一致し、かつ一致したレジスタ番号のレジスタへ結果を格納する演算がまだ完了していない場合に、命令バッファ部201から演算ユニット203への命令フェッチを待たせることに特徴を持つ。

【0055】そこで、命令バッファ部201から信号線202Aを介して1つのVLIWが演算ユニット203にフェッチされ(ST10)演算処理がなされると、演算ユニット203のフィールドiで現在実行中の演算結果が格納されるレジスタ番号(Rj)が信号線204iから取り出され、次に実行しようとしているVLIWに含まれる全ての命令語によって参照されるレジスタ番号が命令バッファ部201Bの信号線202Bから取り出される。信号線204iおよび信号線202Bから取り出されたレジスタ番号は、比較回路205において、一致するかどうかチェックされる。

【0056】比較回路205の4ブロック0~3各々は演算ユニット203のフィールド0~3に対応してお

り、1つのブロックiではそのフィールドiで現在実行中の演算結果が格納されるレジスタ番号と次のVLIW内の命令語によって参照される全てのレジスタ番号との一致が調べられる。一致があれば(ST14, イエス)比較回路205の出力205iは"1"となり(ST16)、不一致ならば(ST14, ノー)出力205iは"0"となる(ST18)。

【0057】比較回路205からの一致出力205iはオア回路207の各ORゲート0~3の一方に入力される。各ORゲート0~3の他方には、演算指定デコーダ206からの出力206iが入力される。

【0058】演算指定デコーダ206は、次に実行しようとしている命令語中で、NOPおよびこの演算が指定されているフィールドiを検出する(ST20)。フィールドiでNOPが検出されると(ST22, イエス)、そのフィールドiに対応するORゲートにロジック"0"の出力206iが与えられる(ST24)。NOPが検出されなければ(ST22, ノー)、出力206iは"1"となる(ST26)。

【0059】出力205iおよび出力206iの何れかが"1"であれば(ST28, イエス)、オア回路207の各ORゲート0~3の出力207iは"1"となる(ST30)。出力207iが"1"というのは、「次に実行しようとする命令語が、フィールドi用のハードウェアリソースを使用する」か、あるいは「次に実行しようとする命令語が、フィールドiで現在実行中の演算結果を参照する」ということを意味する。この出力207iはアンド回路209の対応するANDゲートの一方入力に与えられる。

【0060】一方、フィールドiで現在実行中の演算がその処理サイクル中でまだ終了していないなら(ST32, イエス)、未終了のフィールドiの出力208iが"1"となる(ST34)。この出力208iはアンド回路209の対応するANDゲートの他方入力に与えられる。

【0061】フィールドiについての出力207iおよび出力208iがともに"1"であれば(ST36, イエス)、このフィールドiに対応するアンド回路209のANDゲートの出力209iは"1"となる(ST38)。この出力209iが"1"というのは、「フィールドiで現在実行中の演算が終了するのを待て」ということを意味する。

【0062】アンド回路209の何れかのANDゲート0~3の出力209iが真(ロジック"1")であれば(ST40, イエス)、オア回路210の出力は真(ロジック"1")となる(ST42)。すると、信号線211を介して命令バッファ部201の出力コントローラ201Aに待ち信号(ロジック"1")が伝達され、命令バッファ部201は演算ユニット203へのVLIWの一括転送を停止する。

【0063】フィールド0～3全ての演算が完了すると出力208iは全て“0”になる(ST44, イエス)。するとオア回路207の出力207iに関係なくアンド回路209の出力209iは全て“0”になるから、信号線211の出力も“0”になる(ST46)。すると命令出力コントローラ201Aでの命令転送禁止が解除され、バッファ201B内の次のVLIWが演算ユニット203にフェッチされる(ST48)。

【0064】このようにして、図2に示すような命令語1～4が命令バッファ部201に格納されていてこれらの命令語を順次実行しようとする場合を想定すると、上記実施例のVLIW型演算処理装置では次の処理が実現される。

【0065】すなわち、命令語2と命令語3と命令語4における演算器の参照関係についてコンパイラレベルで考察すれば、命令語2では2つの加算と1つの減算と1つの除算との4演算を行ない、その演算結果をレジスタR1, R3, R4, R8に格納するようにしている。このうち、R1, R3, R4は命令語3で読出されるが、R8は命令語3、命令語4とも参照しない。このR8の中味は除算結果なので、処理に4クロックかかるのではあるが、命令語3と命令語4は必ずしもこの除算の完了を待ち合わせる必要はない。

【0066】命令語3は、2つの加算と1つの乗算の計3つの演算を実行して、R1, R3, R4に結果を格納する。これらのR1, R3, R4は命令語4で読出されるのであるが、とくに乗算の結果が格納されるR4は命令語4で参照される。このため、命令語4は、命令語3の直後の命令であるが、命令語3の乗算の完了を待ち合わせる必要がある。

【0067】したがって、図2に示す命令語1～4を順次処理する場合には、図3に示すように、命令語2の除算を実行している最中に命令語3と命令語4を実行することができる。ただし、命令語4は命令語3の乗算結果を参照しているので、命令語3の乗算結果が出るのを待ってから命令語4を実行することになる。この場合、命令語3と命令語4の実行時間は合わせて3クロック分となるが、これは命令語2の除算と並行して実行されるので、命令語2の完了と命令語4の完了は同時になる。

【0068】こうして、図2に示すような命令語1～4を実行する場合、図3に示すような処理内容と時間経過となる。すなわち、図3では、各命令語1～4の処理時間の総和(所要クロック数)は、図2の場合の8クロックと比べて3クロック少ない5クロックとなる。

【0069】また、図4に示す命令語1～4を順次処理する場合には、図5に示すような処理内容と時間経過となる。図2の場合では命令語4のフィールド3はNOPであるのに対して、図4に示す命令語4のフィールド3は加算を指定している。その他については、図2の命令語の内容と図4の命令語の内容とは同じである。上記命

令語3のフィールド3における内容の違いが処理にどう影響するかについて、以下説明する。

【0070】演算ユニット203では、各フィールド0～3において、加算・減算・乗算・除算・NOPの何れかの演算を行なうことができるが、同時には1種類の演算しか処理できない。そうであれば、図4の命令語4は、命令語2のフィールド3の除算が完了するまでは実行できないことになる。このため、先行する演算を実行中の演算器を後続する命令が使用しようとする場合には、先行演算が完了するまで後続の演算を待たせる制御が必要がある。すなわち、図4の命令語4の実行は命令語2の除算が完了した後に開始されるべきである。このような制御を基に図4に示す命令語1～4が順次処理されると、図5に示すようになり、命令語1から命令語4までの全実行時間は6クロックとなる。

【0071】なお、演算ユニット203の各フィールド0～3において同時に異なった種類の演算が実行できるような構成であれば、図4の命令語4は命令語2の除算と並行して処理できるので、図4の命令列の全実行時間は5クロックで済むことになる。

【0072】以上のように、この発明の実施例でば、先行する演算が実行中のために内容未確定のレジスタを後続命令語が参照しようとする場合と、まだ演算実行中の演算器を使用しようとする場合には、演算が完了して結果が確定するまで後続命令語の実行を待たせる制御を導入しているが、これらの場合以外では、先行する命令語の演算が実行中であっても、原則としてその演算の完了を待たずに後続命令語が実行できるようにして、並列処理能力を向上させている。

【0073】図10は、この発明の他実施例に係るVLIW並列演算処理装置を示す。図10の実施例は、図7の実施例から、次に実行されるVLIW中からNOP命令を検出する構成(演算指定デコーダ206)を省略したのとなっている。

【0074】図10の構成では、VLIWによるオペランドの参照関係をコンパイラレベルで考慮して、以下の制御を行なうことで、フィールド0～3における処理の並列性を確保している。

【0075】すなわち、ある命令語により参照されるオペランド(レジスタ番号)に演算結果を格納しようとするところの、先行起動された演算(例えば乗算)がまだ完了していない場合は、その完了を待ち、そうでない場合は、まだ処理途中の先行起動演算(例えば除算)があっても、この処理途中の演算と並行して後続命令を実行できるような制御を行なう。

【0076】これにより、例えば図3において、フィールド3で除算処理途中にその他の演算処理をフィールド0～2で並列に実行できる。換言すると、VLIWのオペランドフェッチステージでは全てのフィールドに対して命令語を同期して一括投入するが、それらの命令語の

実行ステージではフィールド毎に独立した並列処理を行なうことにより、各フィールドの演算手段（ALU等）の遊びを極力減らす制御が行なわれる。

【0077】

【発明の効果】以上述べたように、この発明では、先行する命令語の演算結果を格納しようとするレジスタ番号が後続の命令語の演算で参照されない場合（または先行命令語演算のために使用中の演算器が後続命令語演算によって使用されない場合）、先行する命令語の演算完了を待たずに後続命令語の並列実行を行なうよう構成したので、並列処理効率が高く、全体として演算処理の高速化が図れる。

【0078】さらに、上記構成において、処理に時間のかかる演算の結果を参照する他の演算の命令語が、前者の演算の命令語から極力離れるようなコードを生成するコンパイラを用意すれば、この発明の装置の処理能力をさらに改善することができる。

【図面の簡単な説明】

【図1】図1は、VLIW命令の一例を示す図。

【図2】図2は、この発明が適用されない場合のVLIW並列演算処理の一例を説明する図。

【図3】図3は、この発明が適用された場合のVLIW並列演算処理の一例を説明する図。

【図4】図4は、この発明が適用されない場合のVLIW

W並列演算処理の他例を説明する図。

【図5】図5は、この発明が適用された場合のVLIW並列演算処理の他例を説明する図。

【図6】図6は、この発明のVLIW並列演算の処理の流れを例示するフローチャート。

【図7】図7は、この発明の一実施例に係るVLIW並列演算処理装置の構成を説明するブロック図。

【図8】図8は、図7の装置によるVLIW並列演算処理の動作を説明するフローチャートの一部。

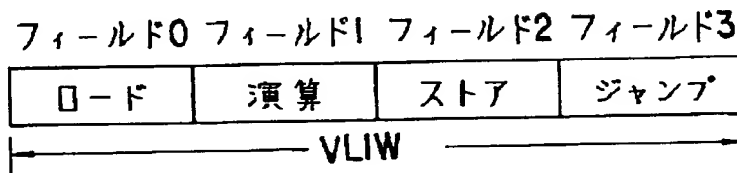
【図9】図9は、図7の装置によるVLIW並列演算処理の動作を説明するフローチャートの他部。

【図10】図7は、この発明の他実施例に係るVLIW並列演算処理装置の構成を示すブロック図。

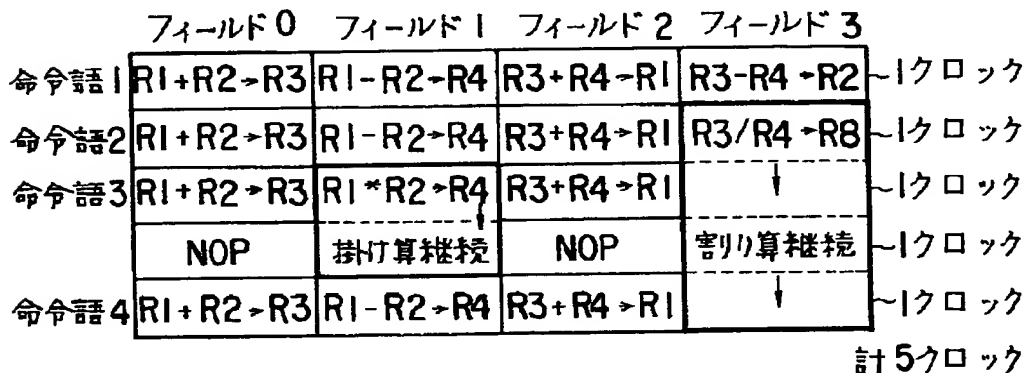
【符号の説明】

200…レジスタファイル（ハードウェアリソース）、201…命令バッファ部（格納手段）、201A…命令出力コントローラ（停止手段）、201B、201C…命令バッファ、203…演算ユニット（実行手段）、205…レジスタ比較回路（提供手段）、206…演算指定デコーダ（実行手段）、207…OR回路、208…信号線（発生手段）、209…AND回路（出力手段）、210…OR回路（出力手段）、205i…一致信号、208i…未了信号、211…命令取り込み信号（“0”）／命令取り込み禁止信号（“1”）。

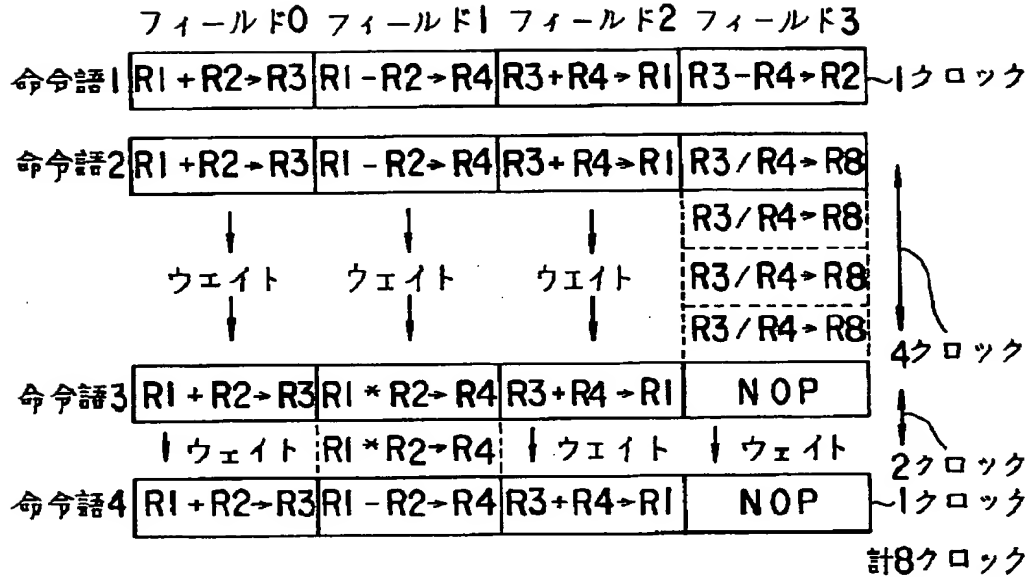
【図1】



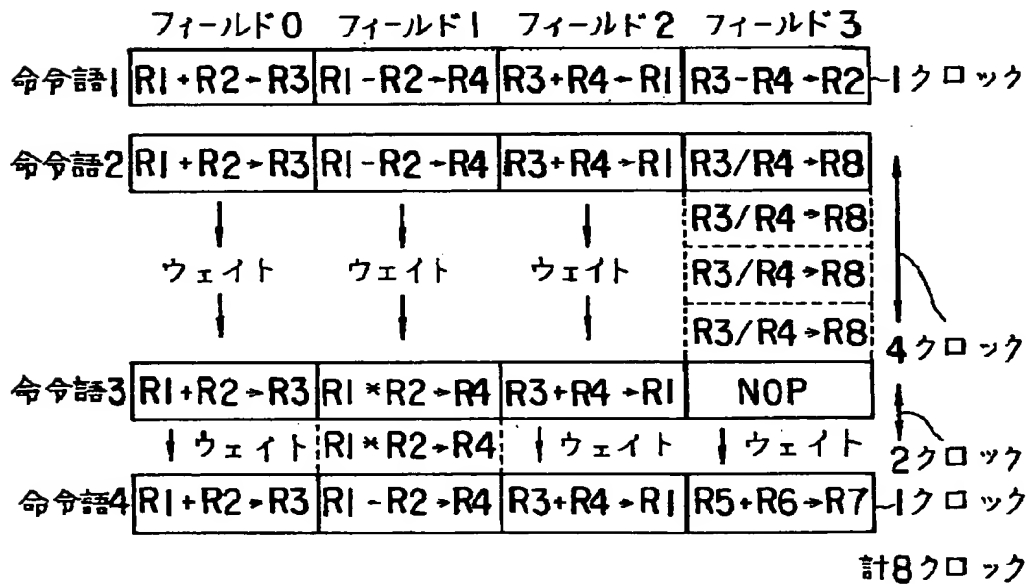
【図3】



【図2】



【図4】

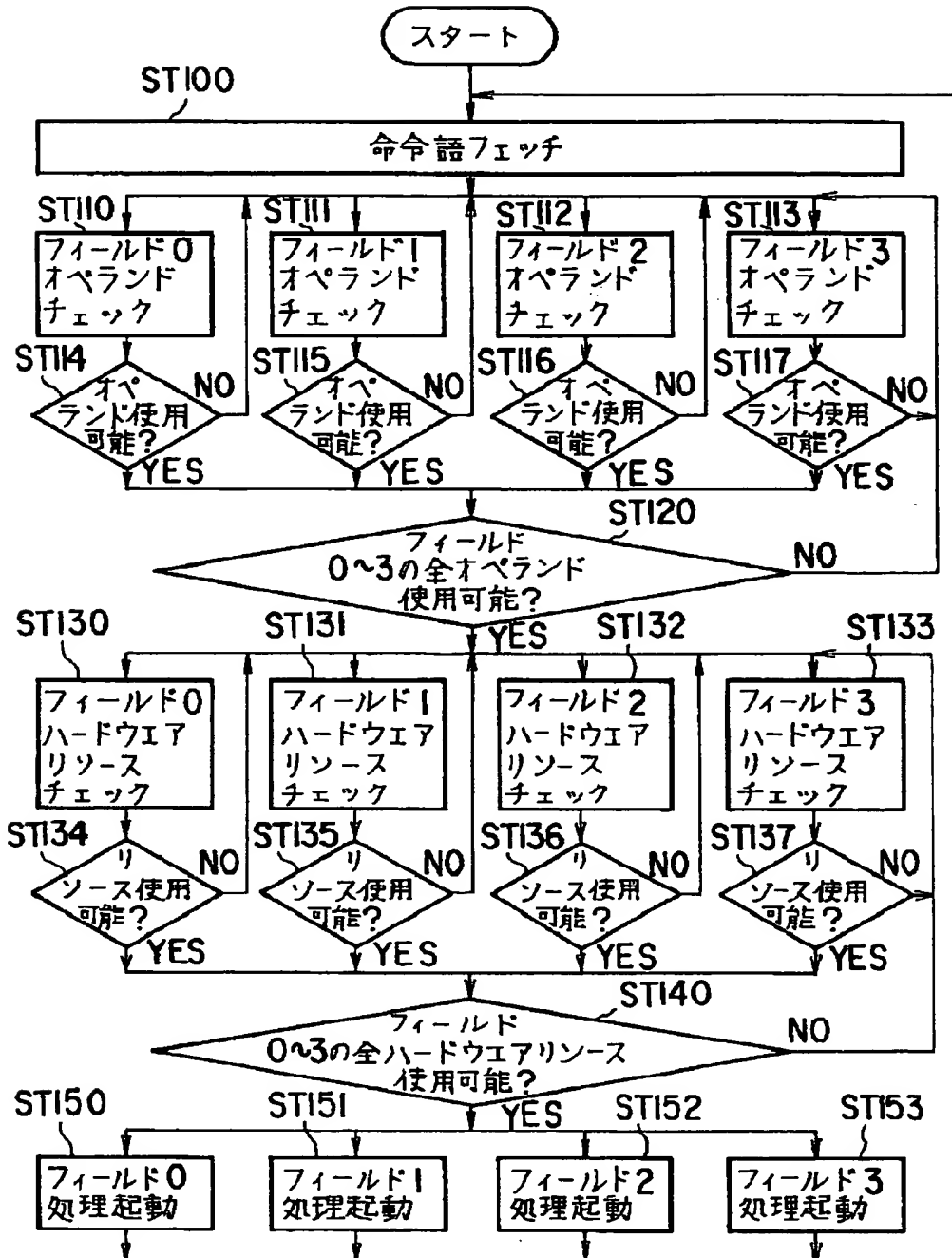


【図5】

	フィールド0	フィールド1	フィールド2	フィールド3	
命令語1	$R1 + R2 \rightarrow R3$	$R1 - R2 \rightarrow R4$	$R3 + R4 \rightarrow R1$	$R3 - R4 \rightarrow R2$	1クロック
命令語2	$R1 + R2 \rightarrow R3$	$R1 - R2 \rightarrow R4$	$R3 + R4 \rightarrow R1$	$R3 / R4 \rightarrow R8$	1クロック
命令語3	$R1 + R2 \rightarrow R3$	$R1 * R2 \rightarrow R4$	$P3 + R4 \rightarrow R1$	↓	1クロック
	NOP	掛け算継続	NOP	割り算継続	1クロック
	NOP	NOP	NOP	↓	1クロック
命令語4	$R1 + R2 \rightarrow R3$	$R1 - R2 \rightarrow R4$	$R3 + R4 \rightarrow R1$	$R5 + R6 \rightarrow R7$	1クロック

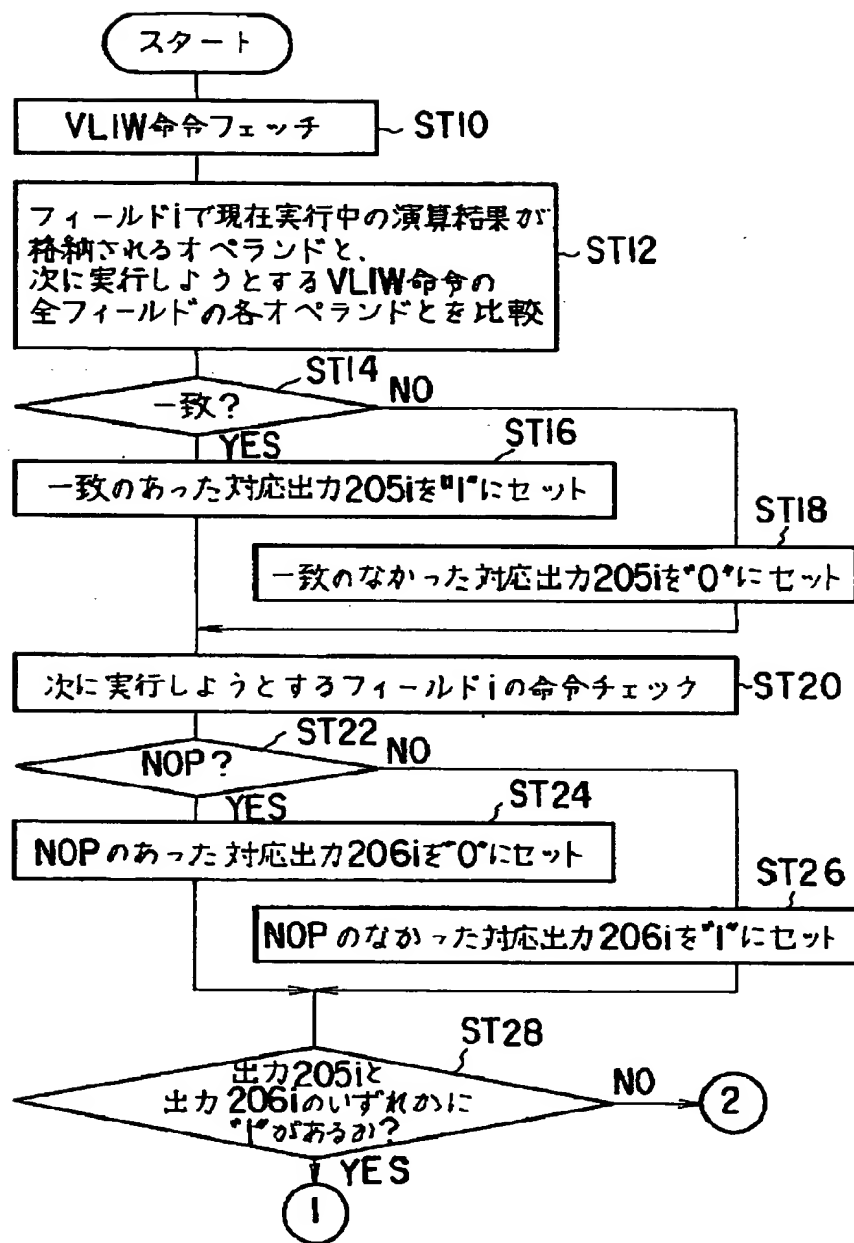
計6クロック

【図6】

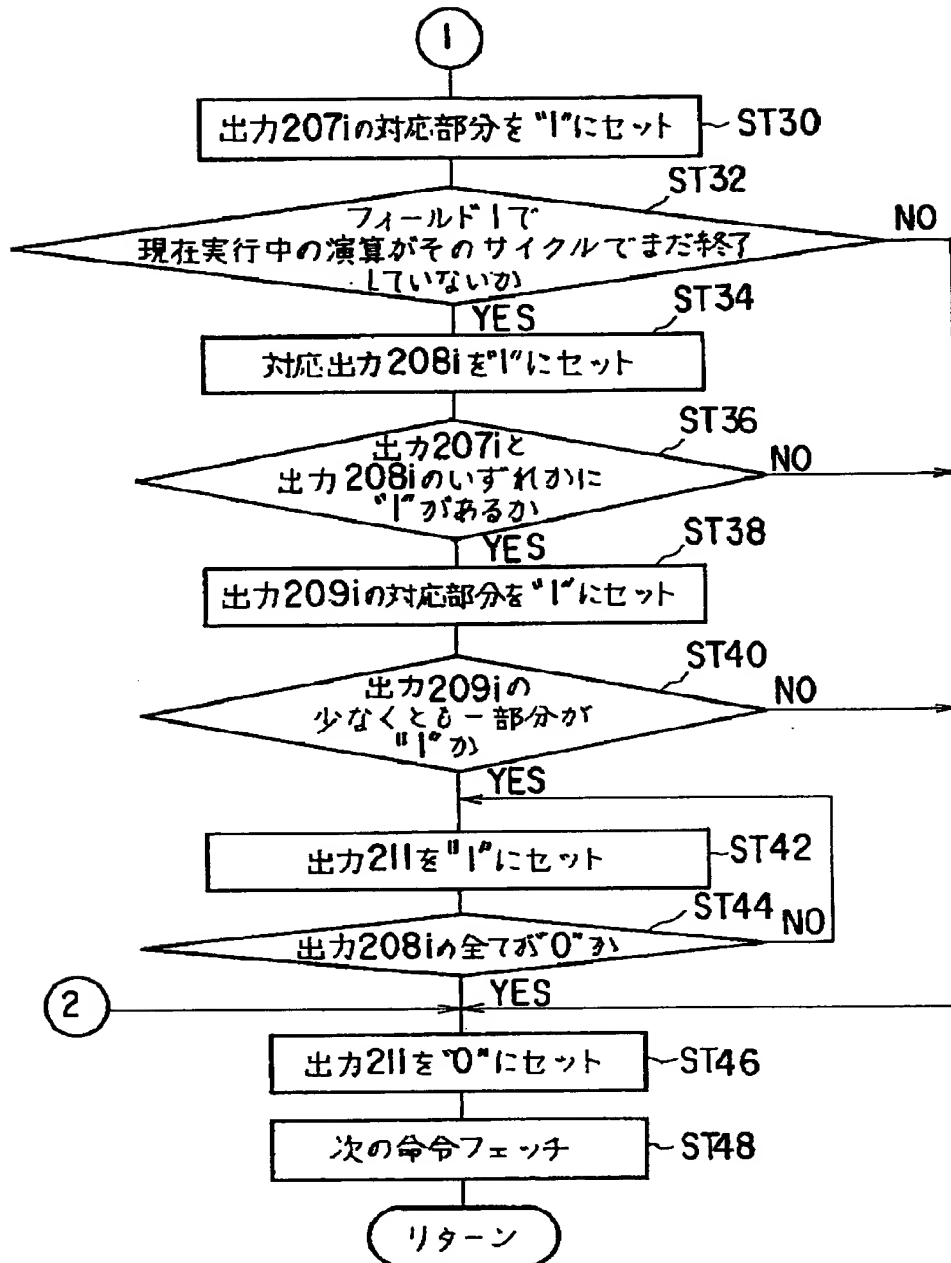


[illegible]

【図8】



【図9】



THIS PAGE BLANK (USPTO)